Proc. Digital del Continguts Musicals

Session 1:
Pattern Recognition

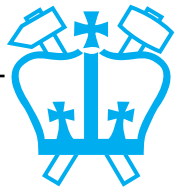
1. **Music Content Analysis**
2. **Pattern Classification**
3. **The Statistical Approach**
4. **Distribution Models**
5. **Singing Detection**

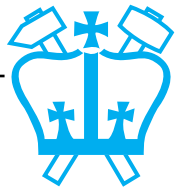
Dan Ellis <dpwe@ee.columbia.edu>
http://www.ee.columbia.edu/~dpwe/muscontent/

Laboratory for Recognition and Organization of Speech and Audio
Columbia University, New York
Spring 2003

1 Music Content Analysis

- **Music contains information at many levels**
  - what is it?
- **We'd like to get this information out automatically**
  - fine-level transcription of events
  - broad-level classification of pieces
- **Information extraction can be framed as: pattern classification / recognition or machine learning**
  - build systems based on (labeled) training data



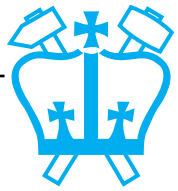


Music analysis

- **What might we want to get out of music?**
- **Instrument identification**
  - different levels of specificity
  - ‘registers’ within instruments
- **Score recovery**
  - transcribe the note sequence
  - extract the ‘performance’
- **Ensemble performance**
  - ‘gestalts’: chords, tone colors
- **Broader timescales**
  - phrasing & musical structure
  - artist / genre clustering and classification

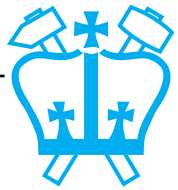
Course Outline

- **Session 1: Pattern Classification**
  - MLP Neural Networks
  - GMM distribution models
  - Singing detection
- **Session 2: Sequence Recognition**
  - The Hidden Markov Model
  - Chord sequence recognition
- **Session 3: Musical Applications**
  - Note transcription
  - Music summarization
  - Music Information Retrieval
  - Similarity browsing



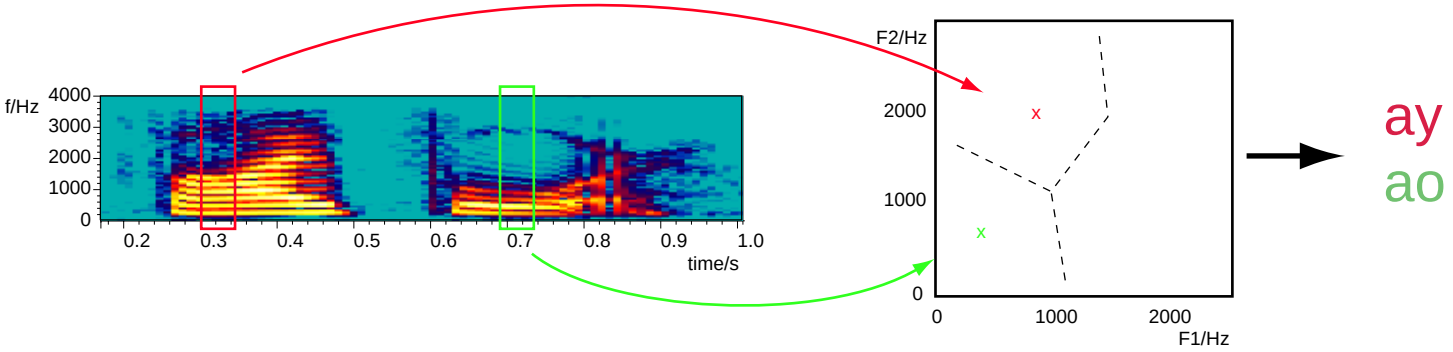
Outline

1. Music Content Analysis
2. **Pattern Classification**
   - classification
   - decision boundaries
   - neural networks
3. The Statistical Approach
4. Distribution Models
5. Singing Detection

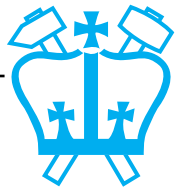


2 Pattern Classification

- **Classification means:**
  finding categorical (*discrete*) labels
  for real-world (*continuous*) observations

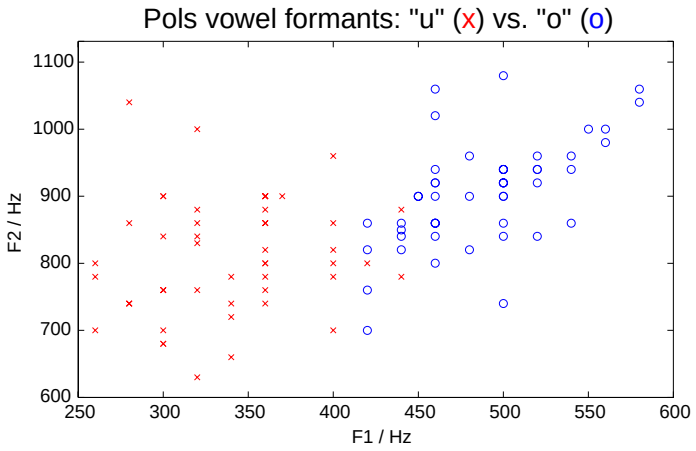


- **Why?**
  - information extraction
  - conditional processing
  - detect exceptions

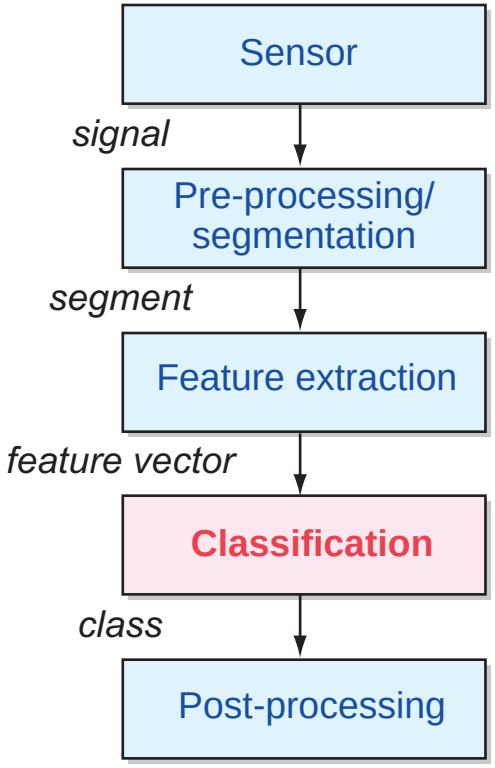


Building a classifier

- **Define classes/attributes**
  - could state explicit rules
  - better to define through ‘training’ examples
- **Define feature space**
- **Define decision algorithm**
  - set parameters from examples
- **Measure performance**
  - calculate (weighted) error rate



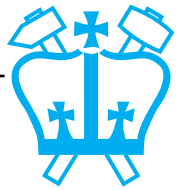
Classification system parts



- STFT
- Locate vowels

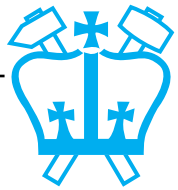
- Formant extraction

- Context constraints
- Costs/risk

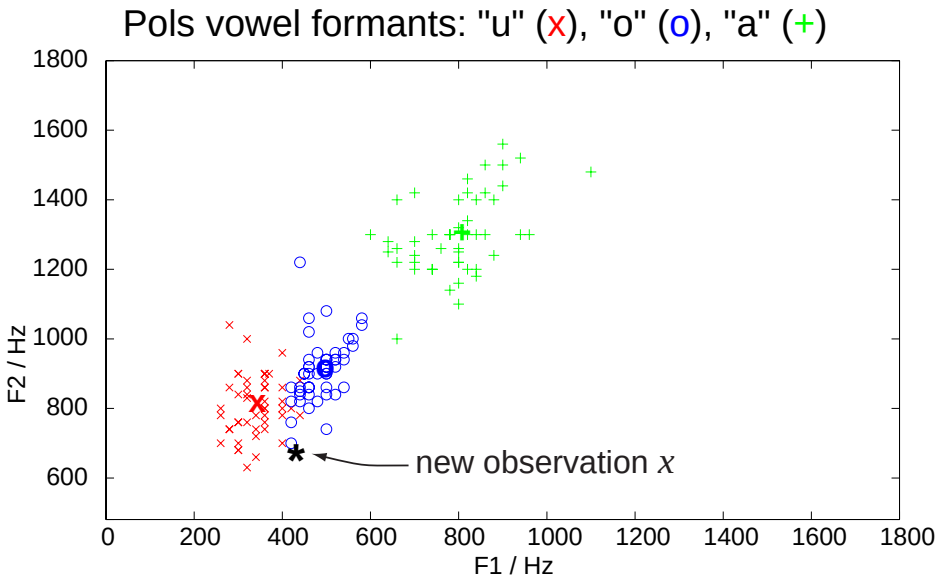


Feature extraction

- **Right features are critical**
  - waveform vs. formants vs. cepstra
  - *invariance* under irrelevant modifications
- **Theoretically equivalent features may act very differently in practice**
  - representations make important aspects explicit
  - remove irrelevant information
- **Feature design incorporates 'domain knowledge'**
  - more data → less need for 'cleverness'?
- **Smaller 'feature space' (fewer dimensions)**
  → simpler models (fewer parameters)
  → less training data needed
  → faster training



Minimum distance classification



- **Find closest match (nearest neighbor)**

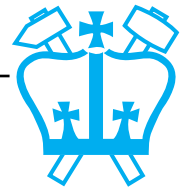
  $min[D(x, y_{\omega,\, i})]$

  - choice of distance metric $D(x,y)$ is important

- **Find representative match (class prototypes)**

  $min[D(x, z_{\omega})]$

  - class data $\{y_{\omega,i}\} \rightarrow$ class *model* z_{ω}



Linear classifiers

- **Minimum Euclidean distance is equivalent to a linear discriminant function:**
  - examples $\{ y_{\omega,i} \} \rightarrow$ template $z_\omega = \text{mean}\{ y_{\omega,i} \}$
  - observed feature vector $x = [x_1\ x_2\ \ldots]^T$
  - Euclidean distance metric

$$D[x, y] = \sqrt{(x-y)^T (x-y)}$$

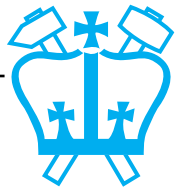
  - minimum distance rule:

$$\text{class } i = \underset{i}{\text{argmin}}\ D[x, z_i]$$

$$= \underset{i}{\text{argmin}}\ D^2[x, z_i]$$

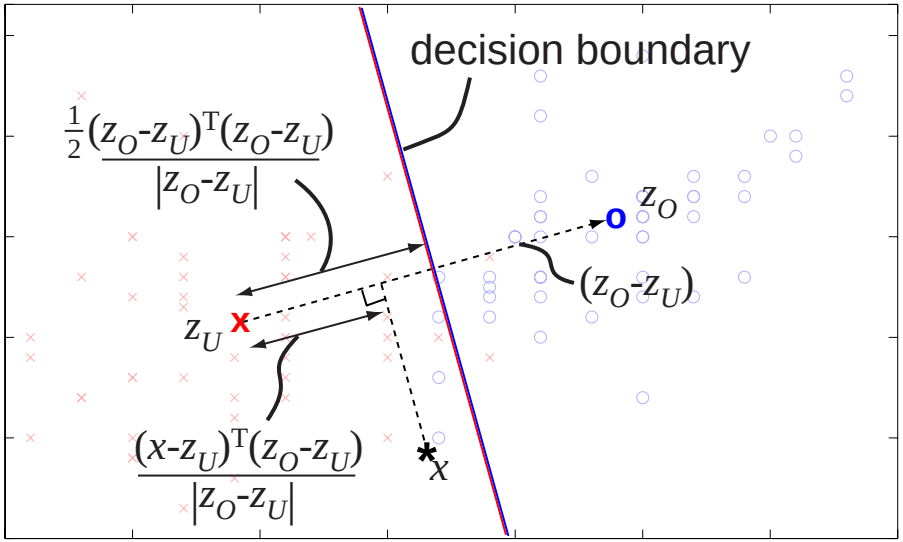
  - i.e. choose class O over U if

$$D^2[x, z_O] < D^2[x, z_U] \quad \ldots$$

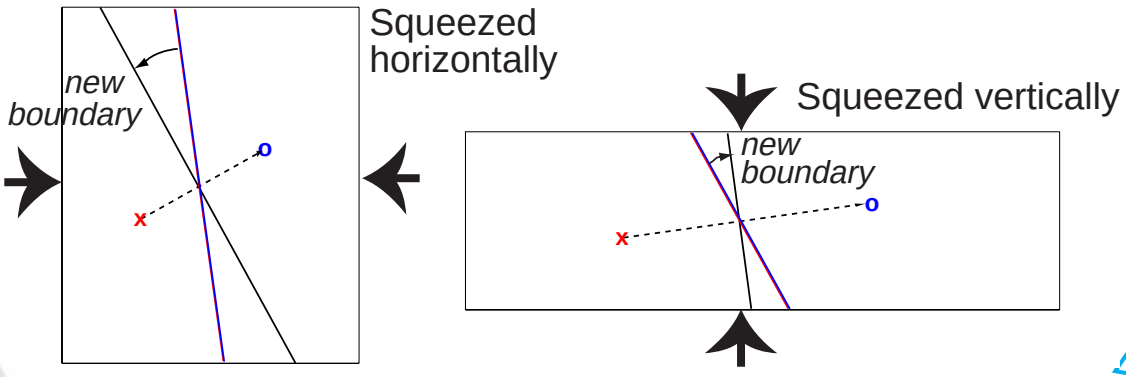


Linear classification boundaries

- **Min-distance divides normal to class centers:**

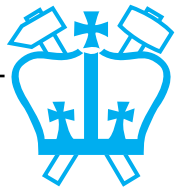


- **Scaling axes changes boundary:**



Decision boundaries

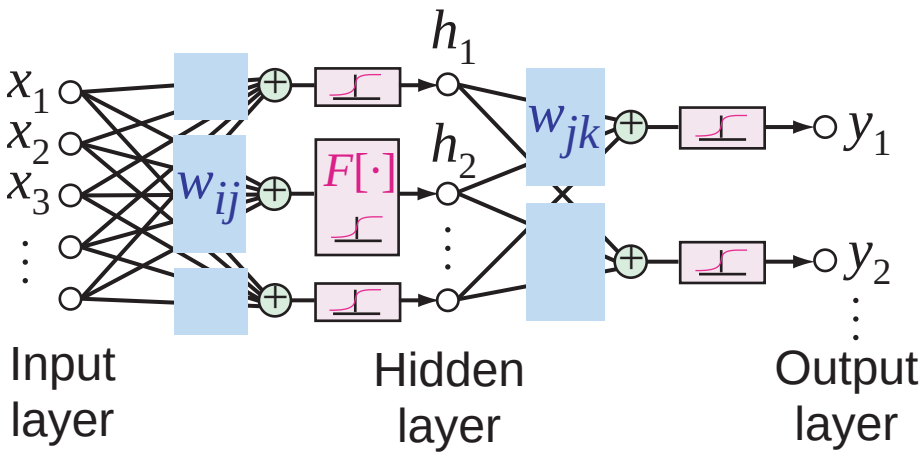
- **Linear functions give linear boundaries**
  - planes and hyperplanes as dimensions increase
- **Linear boundaries can become curves under feature transformations**
  - e.g. a linear discriminant in $\boldsymbol{x}' = [x_1^2\ x_2^2\ ...]^T$
- **What about more complex boundaries?**
  - i.e. if a linear discriminant is $\sum w_i x_i$ how about a nonlinear function?
  - $F[\sum w_i x_i]$ changes only threshold space, but $\sum_j F[\sum_i w_{ij} x_i]$ offers more flexibility, and $F[\sum_j w_{jk} \cdot F[\sum_i w_{ij} x_i]]$ has even more ...



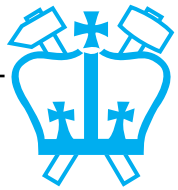
Neural networks

- **Sums over nonlinear functions of sums → large range of decision surfaces**
- **e.g. Multi-layer perceptron (MLP) with 1 hidden layer:**

$$y_k = F[\sum_j w_{jk} \cdot F[\sum_j w_{ij} x_i]]$$



- **Problem is finding the weights w_{ij} ... (*training*)**



Back-propagation training

- **Find w_{ij} to minimize e.g. $E = \sum_n \left| y(x_n) - t_n \right|^2$ for training set of patterns $\{x_n\}$ and desired (target) outputs $\{t_n\}$**
- **Differentiate error with respect to weights**
  - contribution from each pattern x_n, t_n:

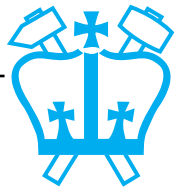
$$y_k = F[\sum_j w_{jk} \cdot h_j]$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial \Sigma} \cdot \frac{\partial}{\partial w_{jk}} (\sum_j w_{jk} h_j)$$

$$= 2(y - t) \cdot F'[\Sigma] \cdot h_j$$

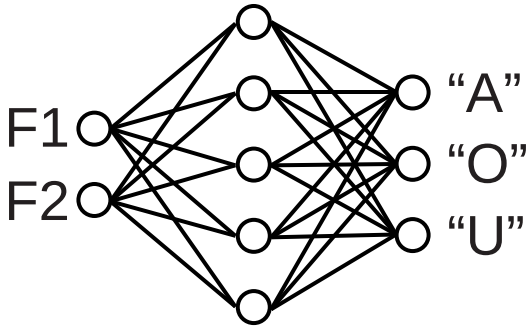
$$w_{jk} = w_{jk} - \alpha \cdot \frac{\partial E}{\partial w_{jk}}$$

- **i.e. gradient descent with learning rate α**



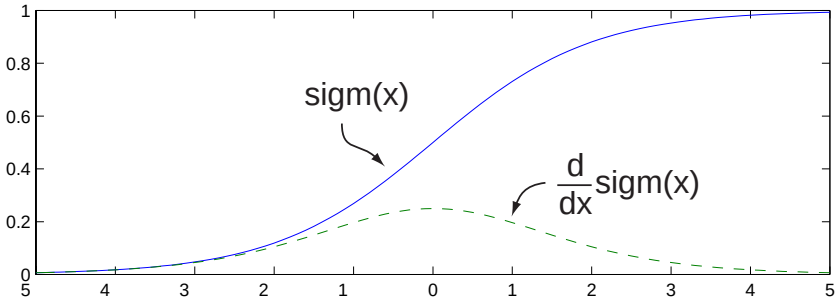
Neural net example

- **2 input units (normalized F1, F2)**
- **5 hidden units, 3 output units (“U”, “O”, “A”)**

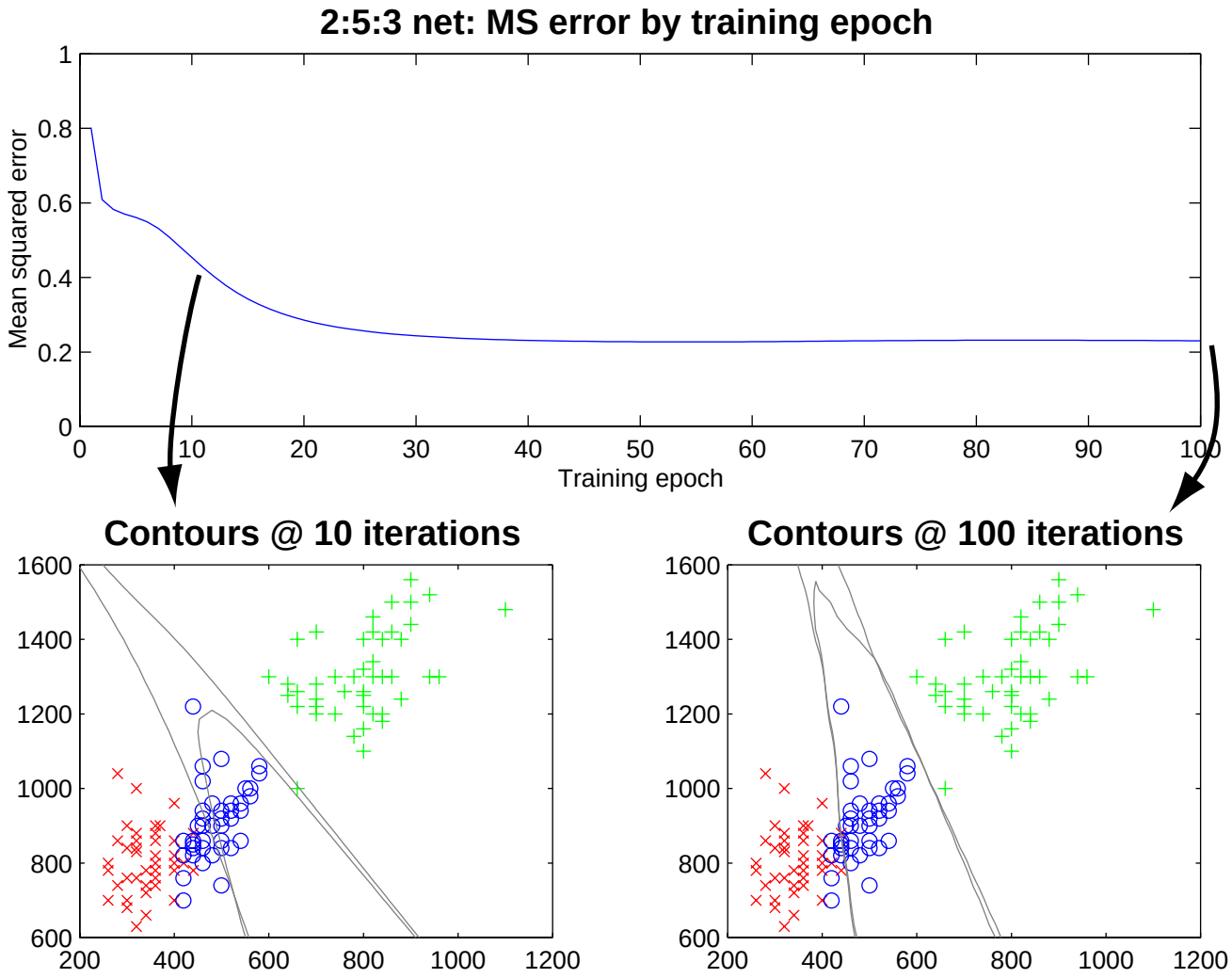


- **Sigmoid nonlinearity:**

$$F[x] = \frac{1}{1+e^{-x}} \quad \Rightarrow \quad \frac{dF}{dx} = F(1-F)$$



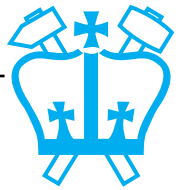
Neural net training



example...

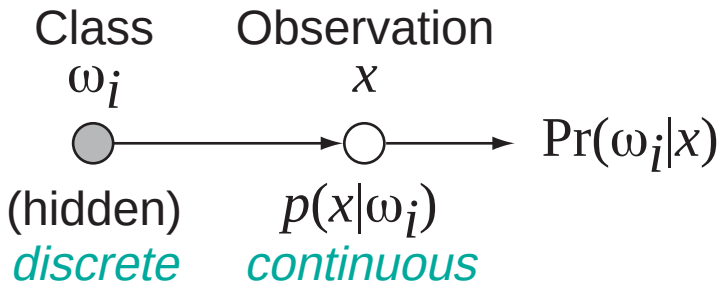
Outline

1. Music Content Analysis
2. Pattern Classification
3. **The Statistical Approach**
   - Conditional distributions
   - Bayes rule
4. Distribution Models
5. Singing Detection

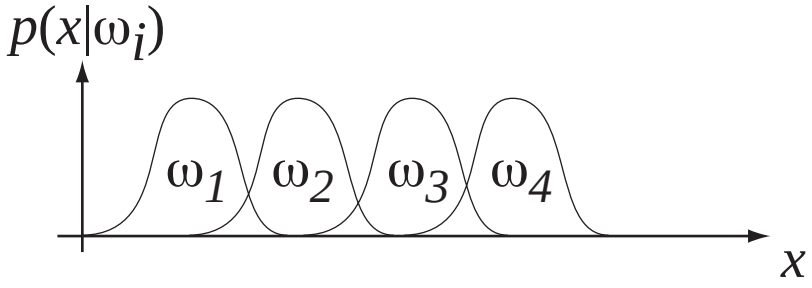


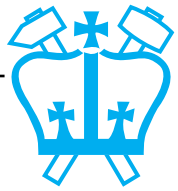
3 The Statistical Approach

- **Observations are random variables whose distribution depends on the class:**



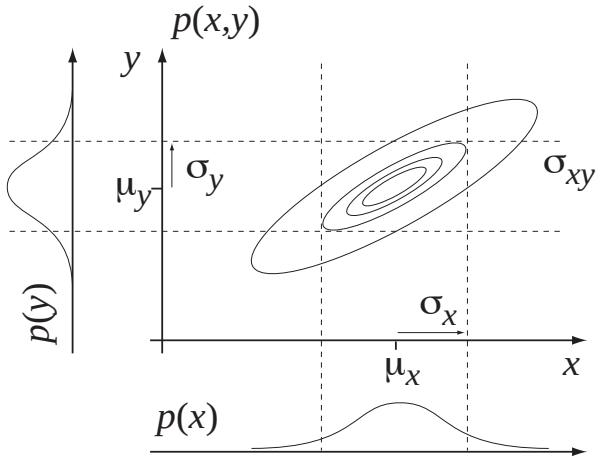
- **Source distributions** $p(x|\omega_i)$
  - reflect variability in feature
  - reflect noise in observation
  - generally have to be estimated from data (rather than known in advance)





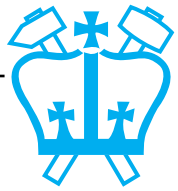
Random Variables review

- **Random vars have joint distributions (pdf's):**



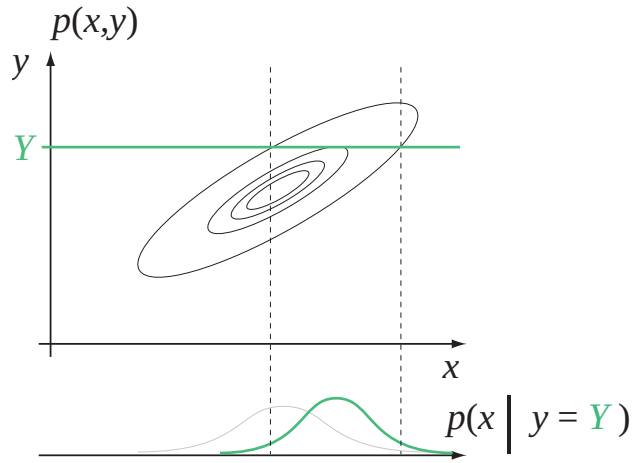
- marginal $p(x) = \int p(x, y)dy$

- covariance $\Sigma = E[(\mathbf{o} - \mu)(\mathbf{o} - \mu)^T] = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$



Conditional probability

- **Knowing one value in a joint distribution constrains the remainder:**

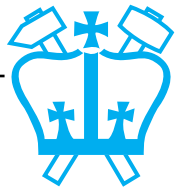


- **Bayes' rule:** $p(x, y) = p(x|y) \cdot p(y)$

$$\Rightarrow p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$

→ **can reverse conditioning given priors/marginal**

- either term can be discrete or continuous



Priors and posteriors

- **Bayesian inference can be interpreted as updating prior beliefs with new information, x:**

Bayes' Rule:

Likelihood

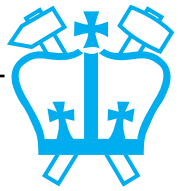
$$Pr(\omega_i) \cdot \frac{p(x|\omega_i)}{\sum_j p(x|\omega_j) \cdot Pr(\omega_j)} = Pr(\omega_i|x)$$

Prior probability

'Evidence' = p(x)

Posterior probability

- **Posterior is prior scaled by likelihood & normalized by evidence (so Σ(posteriors) = 1)**
- **Objection: priors are often unknown**
  - but omitting them amounts to assuming they are all equal



Bayesian (MAP) classifier

- **Minimize the probability of error by choosing *maximum a posteriori* (MAP) class:**

$$\hat{\omega} = \underset{\omega_i}{\operatorname{argmax}} \, Pr(\omega_i | x)$$

- **Intuitively right - choose most probable class in light of the observation**

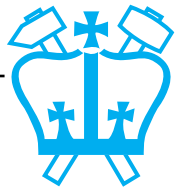
- **Given models for each distribution $p(x|\omega_i)$,**

  the search for $\hat{\omega} = \underset{\omega_i}{\operatorname{argmax}} \, Pr(\omega_i | x)$

  becomes $\underset{\omega_i}{\operatorname{argmax}} \, \dfrac{p(x|\omega_i) \cdot Pr(\omega_i)}{\sum_j p(x|\omega_j) \cdot Pr(\omega_j)}$

  but denominator = $p(x)$ **is the same over all** ω_i

  hence $\hat{\omega} = \underset{\omega_i}{\operatorname{argmax}} \, p(x|\omega_i) \cdot Pr(\omega_i)$



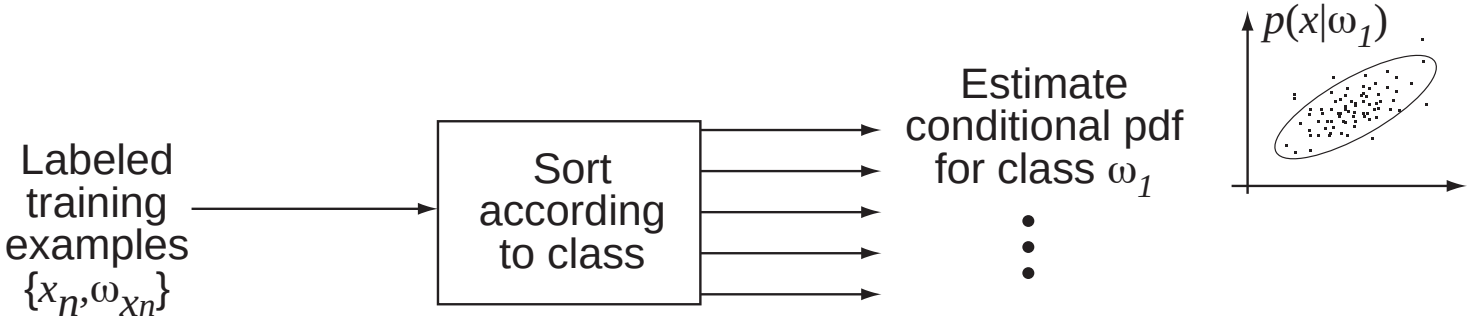
Practical implementation

- **Optimal classifier is** $\hat{\omega} = \underset{\omega_i}{\operatorname{argmax}} Pr(\omega_i|x)$

  but we don't know $Pr(\omega_i|x)$

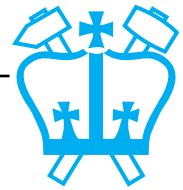
- **Can model directly e.g. train a neural net to map from inputs** x **to a set of outputs** $Pr(\omega_i)$ **- a *discriminative* model**

- **Often easier to model conditional distributions** $p(x|\omega_i)$ **then use Bayes' rule to find MAP class**



Outline

1. Music Content Analysis
2. Pattern Classification
3. The Statistical Approach
4. **Distribution Models**
   - Gaussian models
   - Gaussian mixtures
5. Singing Detection



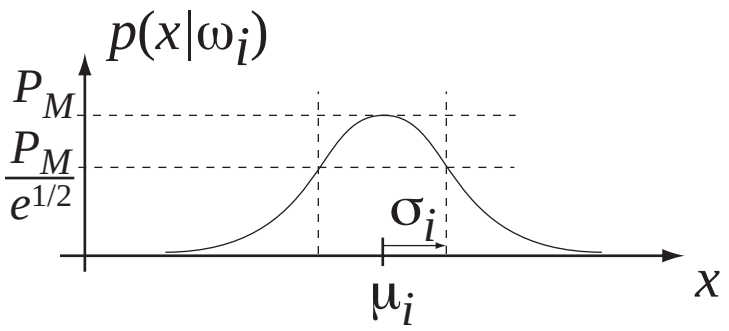
4 Distribution Models

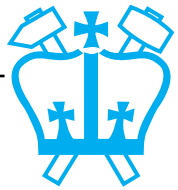
- **Easiest way to model distributions is via parametric model**
  - assume known form, estimate a few parameters
- **Gaussian model is simple & useful:**

in 1 dim: $p(x|\omega_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \cdot \exp\left[-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_i}\right)^2\right]$

normalization to make it sum to 1

- **Parameters mean μ_i and variance σ_i^2 → fit**

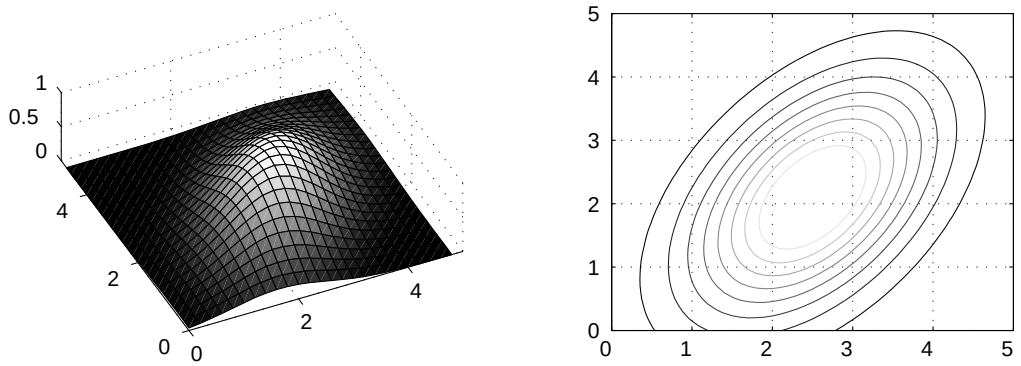




Gaussians in *d* dimensions:

$$p(\mathbf{x}|\omega_i) = \frac{1}{(\sqrt{2\pi})^d |\Sigma_i|^{1/2}} \cdot \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right]$$

- **Described by d dimensional mean vector μ_i and d x d covariance matrix Σ_i**



- **Classify by maximizing log likelihood i.e.**

$$\underset{\omega_i}{\text{argmax}} \left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i) - \frac{1}{2}\log|\Sigma_i| + \log Pr(\omega_i)\right]$$

Gaussian Mixture models (GMMs)

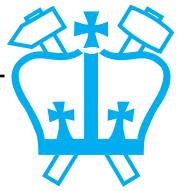
- **Single Gaussians *cannot* model**
  - distributions with multiple modes
  - distributions with nonlinear correlation
- **What about a weighted sum?**

  i.e. $p(x) \approx \sum_k c_k p(x|m_k)$

  where {c_k} is a set of weights and

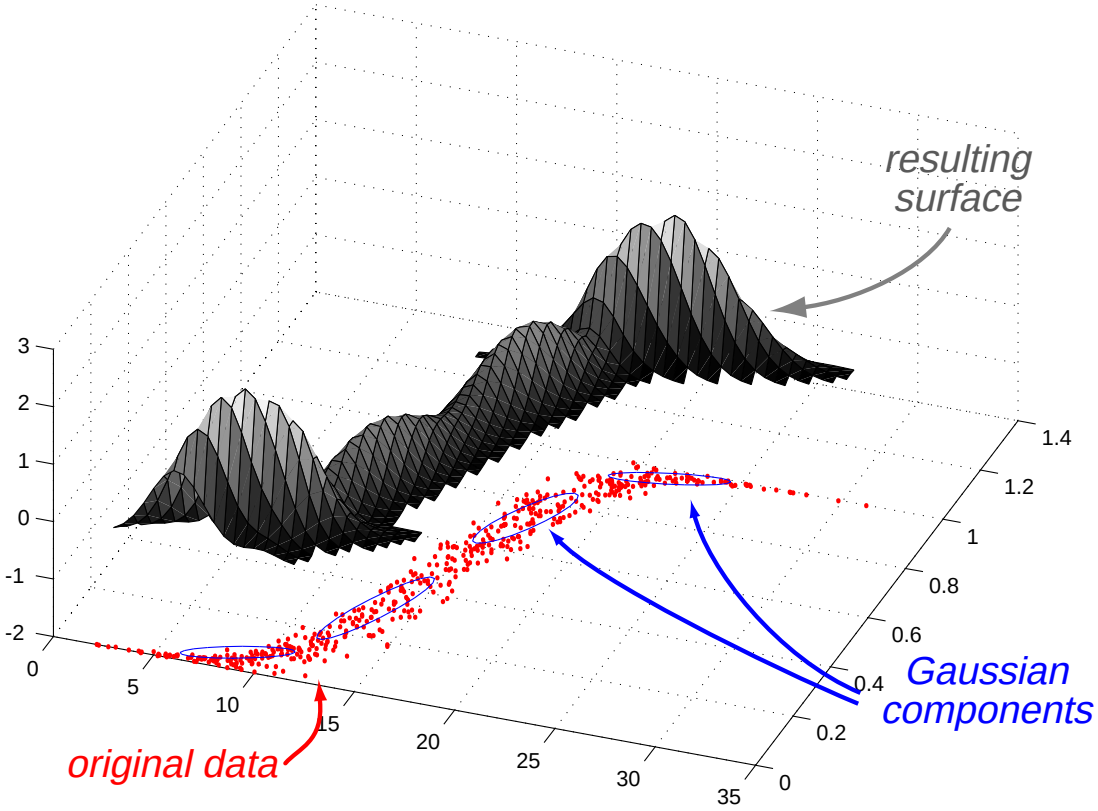
  {$p(x|m_k)$} is a set of Gaussian components

  - can fit anything given enough components
- **Interpretation: each observation is generated by one of the Gaussians, chosen at random with priors** $c_k = Pr(m_k)$

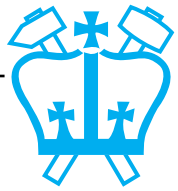


Gaussian mixtures (2)

- **e.g. nonlinear correlation:**



- **Problem: finding c_k and m_k parameters**
  - easy if we knew *which* m_k generated each x

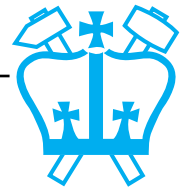


Expectation-maximization (EM)

- **General procedure for estimating a model when some parameters are unknown**
  - e.g. which component generated a value in a Gaussian mixture model
- **Procedure:**
  Iteratively update fixed model parameters Θ to maximize Q=expected value of log-probability of known training data x_{trn}
  and unknown parameters u:

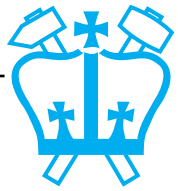
$$Q(\Theta, \Theta_{old}) = \sum_u Pr(u|x_{trn}, \Theta_{old}) \log p(u, x_{trn}|\Theta)$$

  - iterative because $Pr(u|x_{trn}, \Theta_{old})$ changes each time we change Θ
  - can prove this increases data likelihood, hence maximum-likelihood (ML) model
  - *local* optimum - depends on initialization



Fitting GMMs with EM

- **Want to find component Gaussians $m_k = \{\mu_k, \Sigma_k\}$ plus weights $c_k = Pr(m_k)$ to maximize likelihood of training data x_{trn}**
- **If we could assign each x to a particular m_k, estimation would be direct**
- **Hence, treat ks (mixture indices) as unknown, form $Q = E[\log p(x, k|\Theta)]$, differentiate with respect to model parameters → equations for μ_k, Σ_k and c_k to maximize Q...**



GMM EM update equations:

- **Solve for maximizing Q:**

$$\mu_k = \frac{\sum_n p(k|x_n, \Theta) \cdot x_n}{\sum_n p(k|x_n, \Theta)}$$

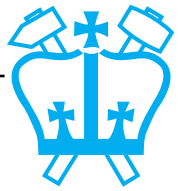
$$\Sigma_k = \frac{\sum_n p(k|x_n, \Theta)(x_n - \mu_k)(x_n - \mu_k)^T}{\sum_n p(k|x_n, \Theta)}$$

$$c_k = \frac{1}{N}\sum_n p(k|x_n, \Theta)$$

- **Each involves $p(k|x_n, \Theta)$,**

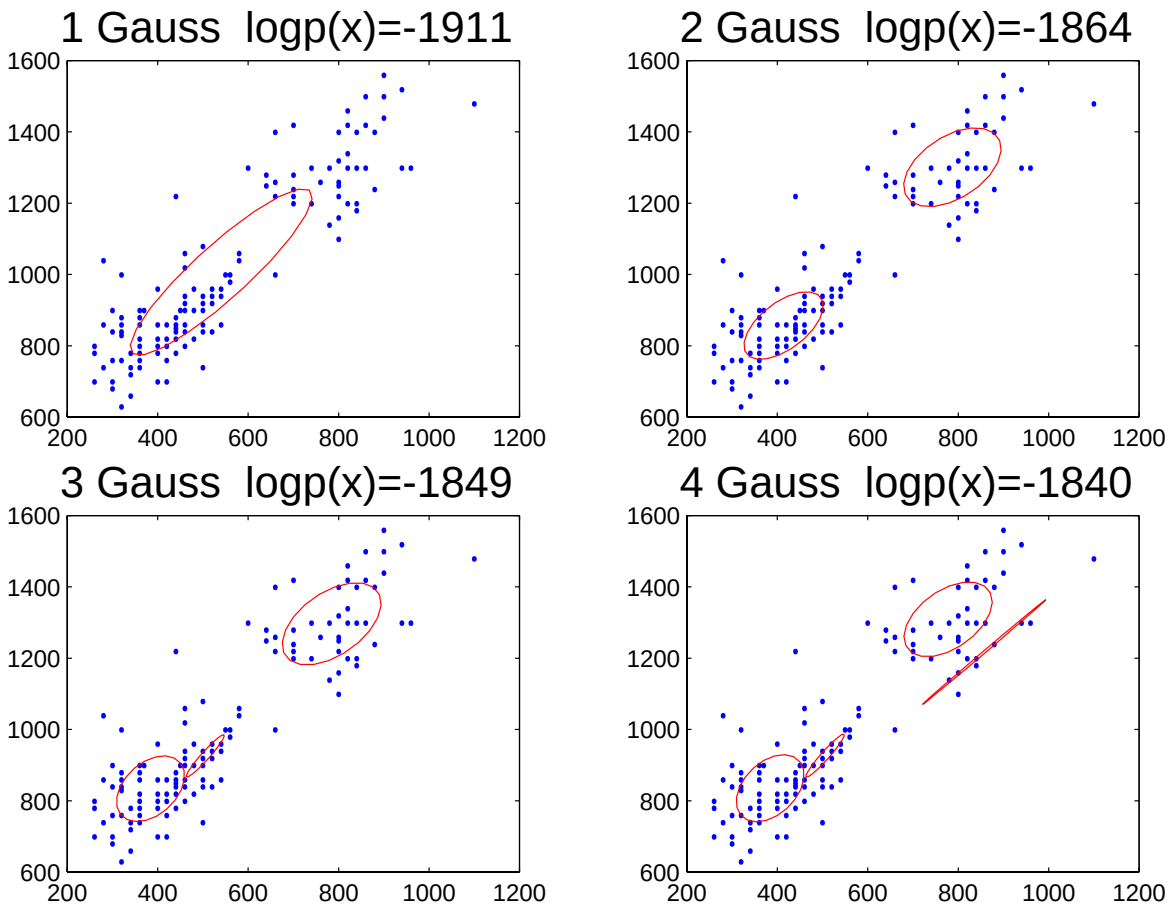
  'fuzzy membership' of x_n to Gaussian k

- **Parameter is just sample average, weighted by 'fuzzy membership'**



GMM examples

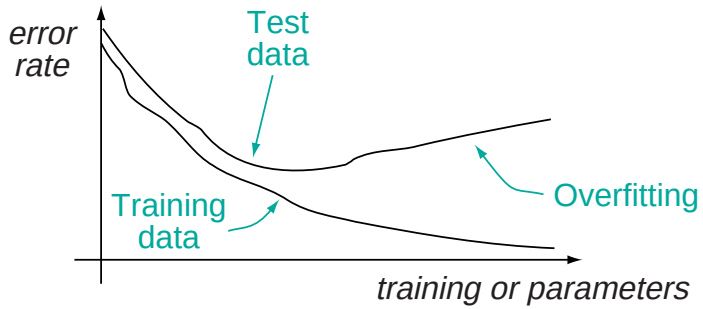
- **Vowel data fit with different mixture counts:**



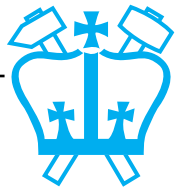
example...

Methodological Remarks: Training and test data

- **A rich model can learn *every* training example (*overtraining*)**

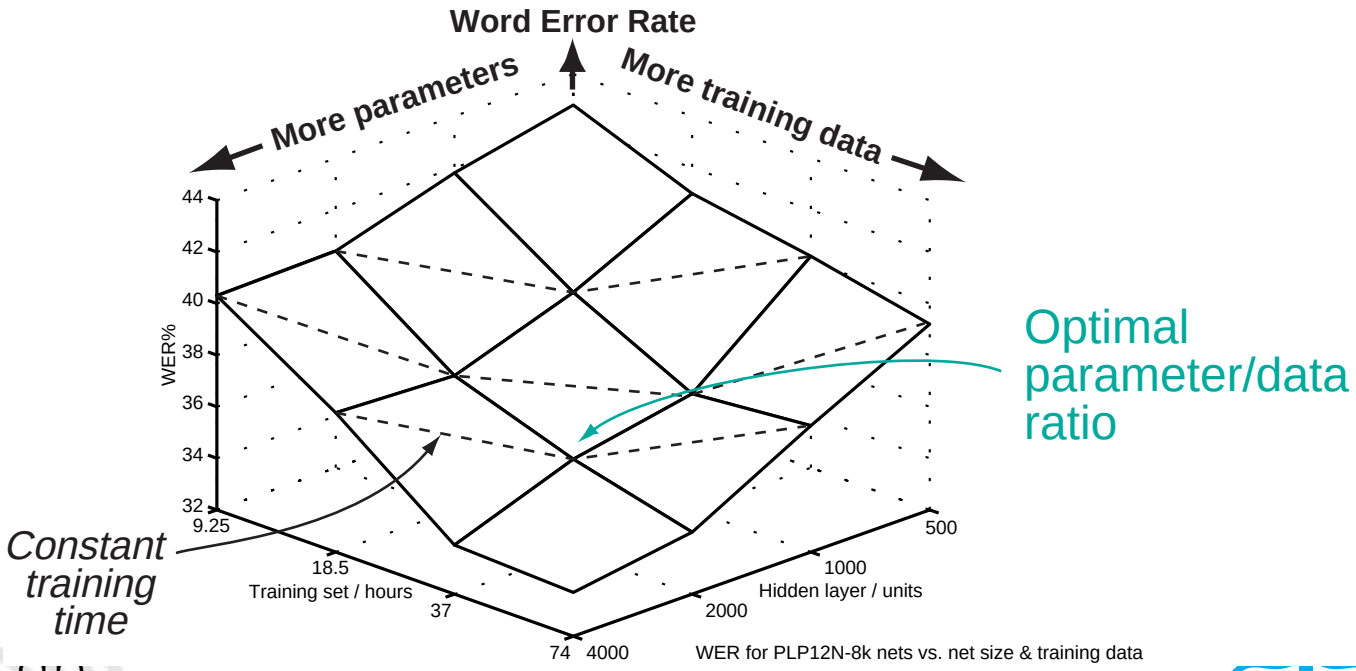


- **But, goal is to classify new, unseen data i.e. *generalization***
  - sometimes use 'cross validation' set to decide when to stop training
- **For evaluation results to be meaningful:**
  - don't test with training data!
  - don't *train* on *test* data (even indirectly...)



Model complexity

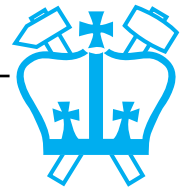
- **More model parameters → better fit**
  - more Gaussian mixture components
  - more MLP hidden units
- **More training data → can use larger models**
- **For best generalization (no overfitting), there will be some optimal model size:**



WER for PLP12N-8k nets vs. net size & training data

Outline

1. Music Content Analysis
2. Pattern Classification
3. The Statistical Approach
4. Distribution Models
5. **Singing Detection**
   - Motivation
   - Features
   - Classifiers

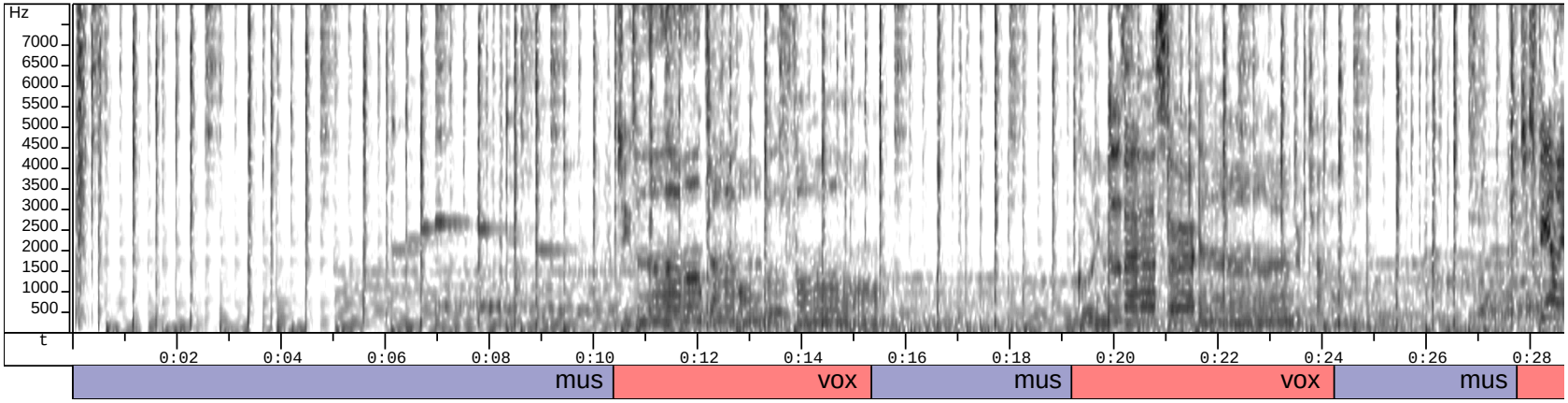


5 Singing Detection

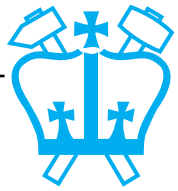
(Berenzweig et al. ’01)

- **Can we automatically detect when singing is present?**

File: /Users/dpwe/projects/aclass/aimee.wav

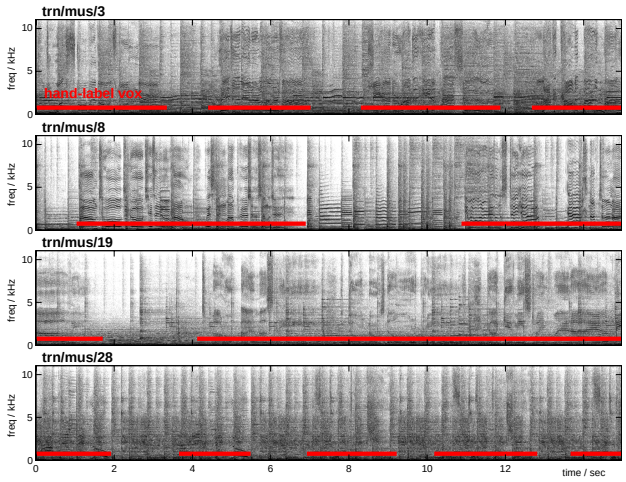


- for further processing (lyrics recognition?)
- as a song signature?
- as a basis for classification?



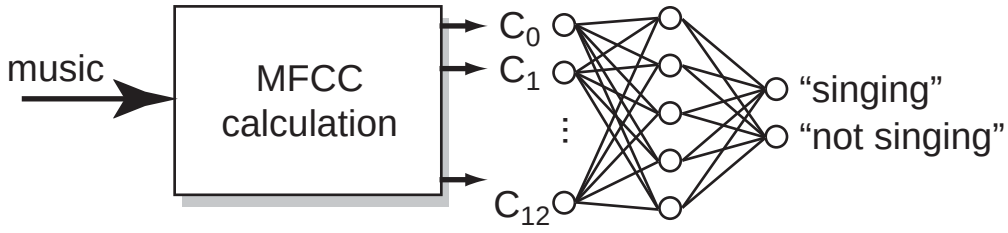
Singing Detection: Requirements

- **Labelled training examples**
  - 60 x 15 sec. radio excerpts
  - hand-mark sung phrases
- **Labelled test data**
  - several complete tracks from CDs, hand-labelled
- **Feature choice**
  - Mel-frequency Cepstral Coefficients (MFCCs) popular for speech; maybe sung voice too?
  - separation of voices?
  - temporal dimension
- **Classifier choice**
  - MLP Neural Net
  - GMMs for singing / music
  - SVM?

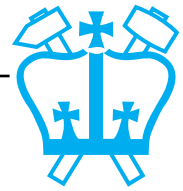


MLP Neural Net

- **Directly estimate** $p(\text{singing} \mid x)$

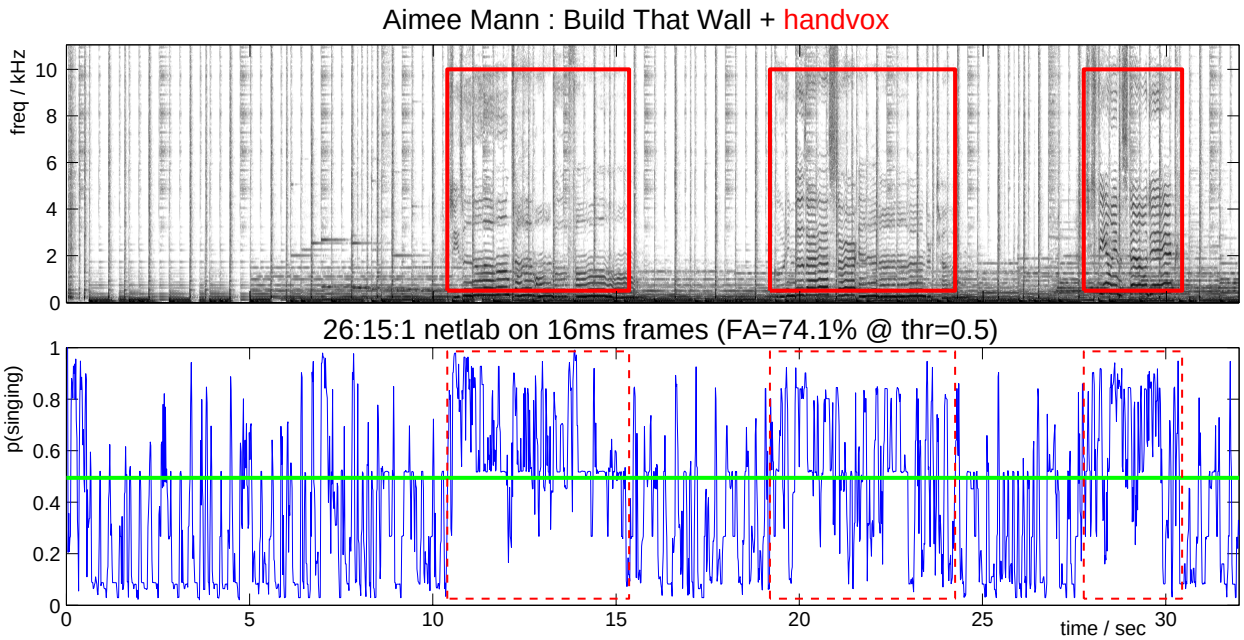


  - net has 26 inputs (+Δ), 15 HUs, 2 o/ps (26:15:2)
- **How many hidden units?**
  - depends on data amount, boundary complexity
- **Feature context window?**
  - useful in speech
- **Delta features?**
  - useful in speech
- **Training parameters...**

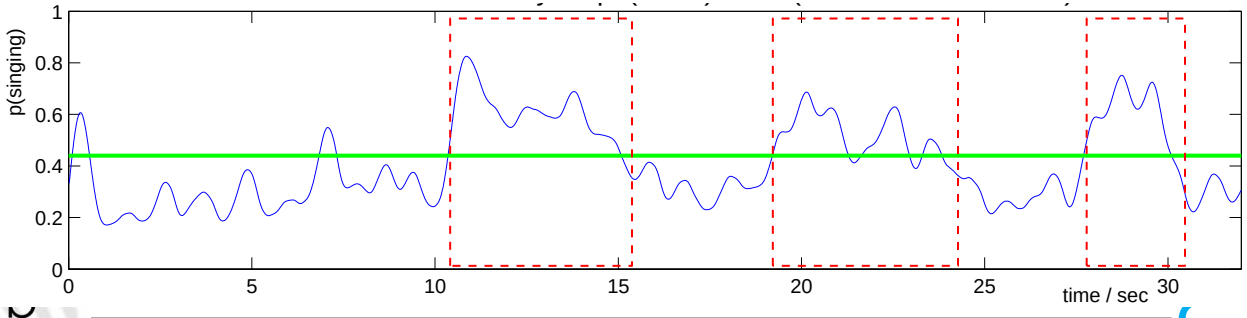


MLP Results

- **Raw net outputs on a CD track (FA 74.1%):**

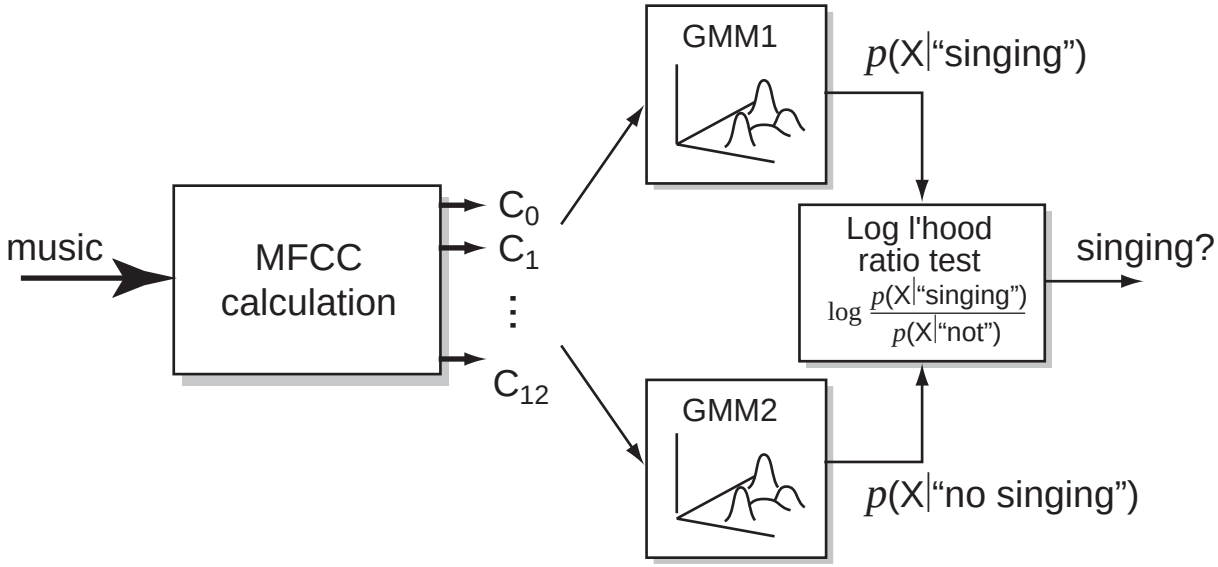


- **Smoothed for continuity: best FA = 90.5%**



GMM System

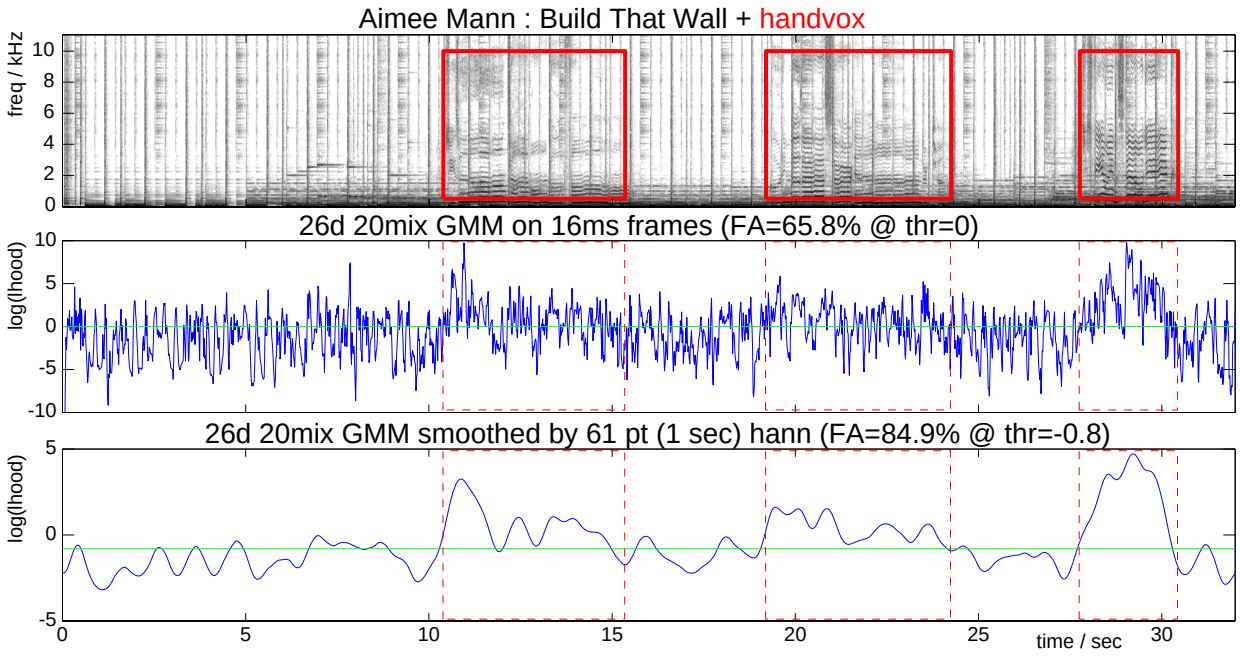
- **Separate models for $p(x|sing)$, $p(x|no\ sing)$**
  - combined via likelihood ratio test



- **How many Gaussians for each?**
  - say 20; depends on data & complexity
- **What kind of covariance?**
  - diagonal (spherical?)

GMM Results

- **Raw and smoothed results (Best FA=84.9%):**

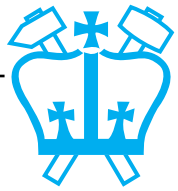


- **MLP has advantage of discriminant training**
- **Each GMM trains only on data subset**
  → faster to train? (2 x 10 min vs. 20 min)

Summary

- **Music content analysis: Pattern classification**
- **Basic machine learning methods: Neural Nets, GMMs**
- **Singing detection: classic application**

but... the time dimension?



References

A.L. Berenzweig and D.P.W. Ellis (2001)
"Locating Singing Voice Segments within Music Signals",
Proc. IEEE Workshop on Apps. of Sig. Proc. to Acous. and Audio, Mohonk NY, October 2001.
http://www.ee.columbia.edu/~dpwe/pubs/waspaa01-singing.pdf

R.O. Duda, P. Hart, R. Stork (2001)
Pattern Classification, 2nd Ed.
Wiley, 2001.

E. Scheirer and M. Slaney (1997)
"Construction and evaluation of a robust multifeature speech/music discriminator",
Proc. IEEE ICASSP, Munich, April 1997.
http://www.ee.columbia.edu/~dpwe/e6820/papers/ScheiS97-mussp.pdf